An Advanced Multi-Element Microcellular Ray Tracing Model

K. H. Ng, E. K. Tameh, A. R. Nix
Centre for Communications Research
Merchant Venturers Building, Woodland Rd,
University of Bristol, Bristol, UK
k.h.ng@bris.ac.uk, tek.tameh@bris.ac.uk, Andy.Nix@bris.ac.uk

Abstract— **In this paper an advanced site specific image-based ray-tracing model is developed that enables multi-element outdoor propagation analysis to be performed in a microcellular environment. Sophisticated optimization techniques such as pre-processing the environment database using object partitioning, visibility determination, diffraction image tree pre-calculation techniques, and parallel processing are used to improve run time efficiency. A comparison of path loss prediction with multi-element site specific measurements shows strong agreement, with a mean error of 3.6dB and a standard deviation of 3.2dB. The model is also shown to be capable of performing detailed MIMO analysis.**

Keywords –ray tracing, propagation, optimisation, MIMO

I. INTRODUCTION

The major drawback of deterministic modelling of the radio propagation channel has been the computational burden compared to that of statistical modelling. However, with the advancement of various modern communication systems, such as the growing use of MIMO [1], in-depth channel characteristic studies in site-specific environments are becoming increasingly important. The Geometrical Optics (GO) based ray-tracing technique has been commonly used in deterministic models and shown to have good agreement with measurement [2]. As with all deterministic models, this method requires a heavy computational load during the ray path finding process.

Section II of this paper describes an overview of our ray tracing model. Section III describes numerous advanced optimization techniques that accelerate the ray path finding process. Section IV compares the prediction of path loss in an outdoor microcellular environment to that of a dedicated set of measurements. Section V describes the MIMO capability of the ray tracing model.

II. OVERVIEW OF RAY TRACING MODEL

Figure 1 shows an overview flowchart of the ray tracing model. Our image based ray tracing model can be divided into four main parts: 1) Pre-processing of database; 2) Creation of the image tree; 3) Creation of the ray tree; 4) Electromagnetic calculation. Pre-processing of the database is used to perform a one-time optimization of the environment database to accelerate the ray path finding process during run time. The ray path finding process is used to find all possible ray paths from the transmitter source to the receiver. This task includes the forward creation of the image trees and the creation of the ray tree through backward tracing. Our ray tracing model is able to find extensive ray paths that capture major propagation mechanisms, such as building reflection, building roof top diffraction, building corner diffraction, building scattering and terrain scattering (and combinations of these various propagation mechanisms). These ray paths are calculated in full 3D. A vertical plane diffraction model is also supported in our tool to approximate multiple building roof top diffractions for faster processing. The capability of our model to combine these key propagation mechanisms allows comprehensive analysis to be performed in an outdoor microcellular environment. The electromagnetic calculation stage applies various EM models to the generated ray paths, these include GO Fresnel for reflection, Uniform Theory of Diffraction (UTD) for diffraction, and synthetic aperture radar techniques for terrain scattering [3][4].

The 3D object geometry used in the ray tracing model consists of polygons, polygon tiles, polygon horizontal edges, polygon vertical edges and terrain maps. This is similar to the definitions found in [5]. The difference here is that in [5], edges are divided into segments and all interactions (reflection and diffraction) occur at the centre of every visible tile and segment. In the ray tracing model discussed here, the visible centre of polygon tiles are used for scattering, polygons are used for reflection, polygon horizontal and vertical edges are used for diffraction and terrain maps are used for scattering. This avoids the need to break down the polygons into segments and tiles for reflection and diffraction purposes due to the nature of image-based ray tracing, and hence reduces the number of interaction objects. For example, when considering reflection from one potential visible polygon surface, [5] has to perform calculations at every visible centre of the tiles, whereas in our approach only one reflection calculation is needed.

III. OPTIMIZATION OF RAY TRACING MODEL

A number of advanced acceleration techniques have been implemented, including object space partitioning, visibility determination, pre-creation of edge diffraction trees and grid computing. Some of these techniques have greatly enhanced the efficiency of the ray tracing process [4-6]. These techniques can be performed during the database pre-processing stage.

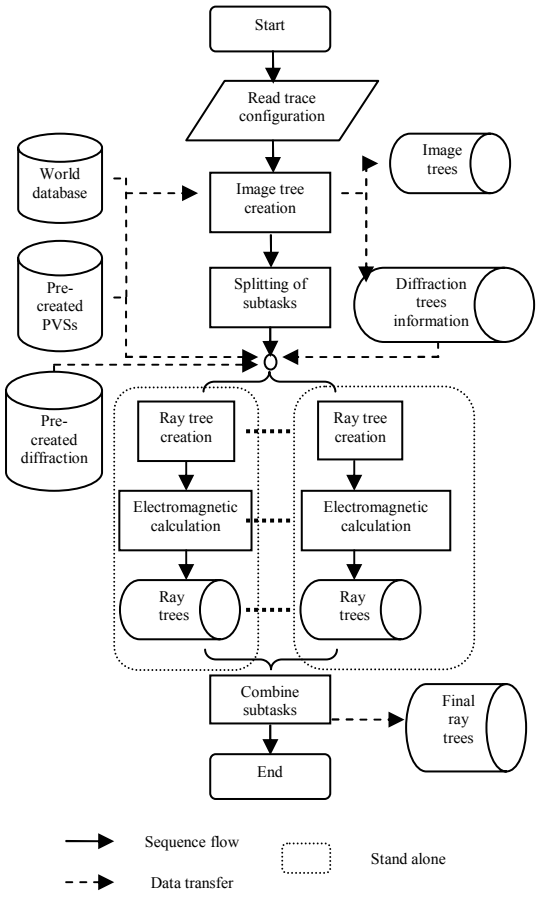

Figure 1. Flow chart of ray tracing model.

A. Object Space Partitioning

Object space partitioning is an effective spatial partitioning method for creating powerful data structures that enable fast object spatial handling [8]. For databases with many objects, object space partitioning can greatly accelerate spatial operations such as proximity queries, ray casting and ray intersections. In object space partitioning, objects are hierarchically partitioned into various spatial cells, called 'leaves', according to a set of partitioning 'planes'. Each set of 'planes' form a 'node'. The leaves and nodes form a 'tree' structure that allows fast searching based on spatial coherency. An object space partitioning data structure is named according to the number of 'planes' in each 'node'. A binary tree, quadtree and octree require two, four and eight 'planes' respectively.

A hybrid intra-object BSP tree and inter-object quadtree data structure is implemented here for the ray tracing model. Each object (building and foliage) is partitioned into individual BSP trees. All the BSP trees in turn are partitioned into a single quadtree. The reason for using a hybrid spatial partitioning method arises from the fact that the use of a BSP in a complicated outdoor environment creates many implementation problems (especially floating point errors) and a volume bounding quadtree does not include information about the polygons in each object. Since the BSP tree is already constructed for each object for Constructive Solid Geometry purposes (i.e., for building a 3D world from the raw database) [8], the hybrid method is feasible to combine the powerful features of both a BSP and quadtree. This is different from [5-7], where only one type of object space partitioning has been implemented.

In order to see the improvement for line of sight tests using object space partitioning, a comparison was performed using 1,000,000 sets of randomly generated start and end locations using the database described in Section IV. Three types of algorithm were examined. The first uses a simple ray box and ray plane intersection method, which checks against each object using a brute-force method. The second method uses the same testing algorithm except the data is now arranged in a quadtree. The third method uses the hybrid technique discussed previously. The results are shown in Table 1 below[*]. It can be seen that object space partition using the hybrid method improves the processing time by 192% and 44% compared to the simple intersection and quadtree methods respectively.

TABLE I. PROCESSING TIME FOR THREE DIFFERENT LOS ALGORITHMS.

Method	Simple Intersection	Quadtree	Quadtree +BSP
Processing Time (s)	152	75	52

B. Visibility Determination

One way to improve the efficiency of a ray tracing algorithm is to reject objects early when ray intersection is impossible. The result of visibility determination can be stored in a data structure known as the Potential Visibility Set (PVS). A PVS is a set of potential visible information [9]. It is basically a table of simple 'Yes' or 'No' entries on object visibility. For ray tracing purposes, it is important to have a set of PVSs for inter-object and point-object visibility. Inter-object PVSs allows fast visibility determination for rays between objects. Point-object PVSs determine the visibility of objects from emitter points. These PVSs are compressed using simple zero-run-length coding [9] for optimal storage and fast access during run-time.

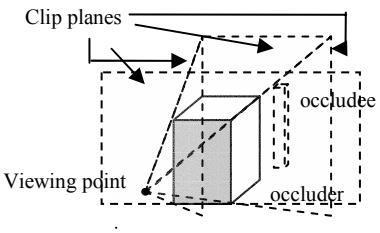

Figure 2. Occlusion culling.

The simplest form of visibility determination is back-face culling. Back-face culling is a method to cull away object geometry that is back-facing to the viewer. It is a simple algorithm that uses a vector dot product operation to determine the facing orientation. Hence it provides the fastest crude way to eliminate non-visible objects. A more effective form of visibility determination makes use of object occlusion culling

[*] All computing nodes are based on Windows Pentium-4 2.4GHz platform.

techniques [8]. This is a method that eliminates not only back-facing object geometry, but also any occluded object geometry. Occlusion occurs when the object geometry is blocked by other object geometry. The occludee is the occluded object geometry and the occluder is the occluding object geometry (see Figure 2). For the environment used here, the occludees include polygon horizontal edges, polygon vertical edges, polygon tile centres and terrain map centres. Occluders consist of a set of clip planes that enclose the shadow region. The culling process can be simplified by clipping the occludee against the clip planes. Practically, the clippling process can be performed using a memory filling method, where a buffer representing the physical layout of the geometry is created and the buffer representation of the geometry that falls within the shadow region is bit-invalidated.

The crucial part of occlusion culling is to create a set of occluders that contain information on the shadow regions so that the visibility of the occludee can be checked. The occluders are created in the following manner: (i) Arrange all polygons in order of distance from the source to create a list of occludees. (ii) Choose the nearest polygon in the occludee list to be an occluder; create a set of clip planes from the source to the polygon; perform occlusion culling on the rest of unoccluded polygons in the occludee list; remove the occluded polygons from the occlude list. (iii) Each unoccluded polygon that remains in the occludee list is removed from the list and becomes an occluder in turn. Process (ii) is repeated until no polygons are left in the occludee list.

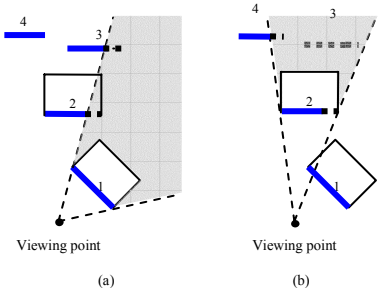

Figure 3. Occlusion culling process.

Four potential occluders are shown in Figure 3 as thick solid lines. The shaded area is the shadow region formed by the clipping planes. All objects are higher than the viewing point. From Figure 3a, during the first occlusion culling for the first occluder, parts of the second and third potential occluders are occluded as represented by the thick dotted lines. These correspond to the zero filling of the visibility bit sets. From Figure 3b, since the second potential occluder is not fully blocked, it becomes an occluder. Occlusion culling is performed for the second occluder. As a result, the third potential occluder is now fully blocked and hence discarded (not visible). A fourth potential occluder is only partly blocked and would next become an occluder.

Figure 4 shows the complementary cumulative density function (CCDF) of the percentage visibility reduction between the simple back-face culling method and the more detailed occlusion culling method. The visibility reduction is calculated from the difference in the number of visible elements for the two schemes as a percentage of the total number of elements. The calculation is performed on the database described in Section IV for four different PVSs: 1) Polygon to polygon; 2) Vertical edge to vertical edge; 3) Polygon to terrain; 4) Vertical edge to terrain. It can be seen from Figure 4 that the visibility determination with occlusion culling reduces the number of visible elements significantly when compared to simple back face culling, with an average reduction in the number of visible elements of 21%, 57.5%, 48.2% and 76.2% for each of the four PVSs. A reduction in visibility results in an improvement in the speed of the ray tracing process as less object interactions are performed. Figure 5 shows the potential visible front facing polygons (shaded white) from a given view point, S.

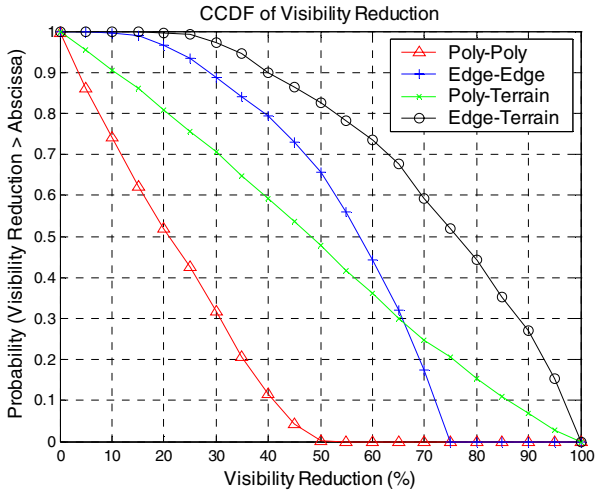

Figure 4. CCDF of visibility error as a percentage for four different PVS calculations using occlusion culling and simple back face culling.

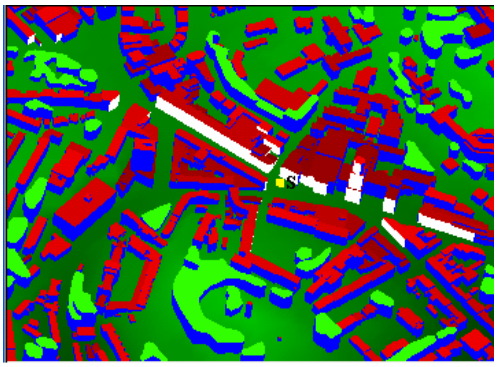

Figure 5. Point to polygon visibility.

C. *Pre- creation of Diffraction Trees*

For image-based ray tracing, generating images and storing the image trees is computationally costly. An image tree is formed based on knowledge of the source position and an object database. The source can be either a transmitter or a vertical diffraction edge [2]. The overall image tree for each ray-tracing run consists of one transmitter image tree with branches of diffraction image trees at each visible diffraction

edge. As all diffraction edge source positions are static for each environment database, generating diffraction image trees for every ray-tracing run is redundant. Therefore, it is possible to pre-create all the diffraction image trees for each database. These pre-created diffraction image trees are dynamically linked at run-time. One constraint of this technique is that the maximum order of reflection after diffraction for each ray is limited by the order of the static diffraction image tree (as with all image trees). Nevertheless, if a high order of reflection after diffraction is needed, it is possible to expand the pre-created diffraction tree at run time at the expense of speed.

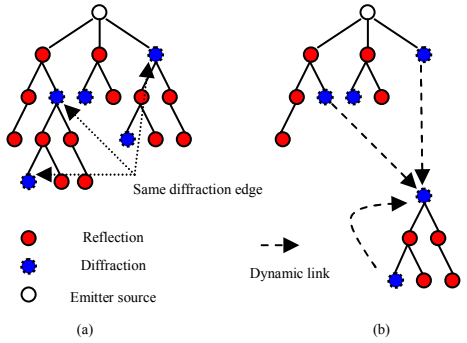

Figure 6. Pre-creation of diffraction trees.

Figure 6a shows an image tree with solid shaded circles representing reflection and dash shaded circles representing diffraction. Three of the diffraction nodes have the same diffraction edge and hence they have the same child tree. Figure 6b shows that this redundancy is removed by dynamic linking to a pre-created diffraction tree.

D. Grid computing

One advantage of image-based ray tracing is that the same image tree can be used in the creation of all ray trees with the same trace configuration. A typical ray tracing application has many receiver points (ray trees) and this provides a way to exploit parallel processing. The ray tracing model discussed here can split the creation of the ray trees into multiple subtasks, each handling a number of receiver points. Each subtask is capable of performing stand-alone ray tracing with the same image tree. Hence, the whole ray tracing process can be distributed into multiple subtasks for the grid computing process (see Figure 1). This greatly enhances the speed of the ray tracing process and makes it more feasible to perform rapid route and grid analysis. The only insignificant overhead incurred is the need to combine subtask results into a single structure. The speed of ray tracing can be improved linearly with the number of grid computing processors.

Table 2 shows a comparison of the processing time needed in the ray tracing trial described in Section IV, for the case of a single processor and a grid with 10 processors. It can be seen here that grid computing has reduced the overall processing time by around 307% as compared to a single processor. By comparing the time of creation of ray tree, the use of grid computing has actually reduced run-time by around 850%.

TABLE II. PROCESSING TIME FOR SINGLE PROCESSOR AND GRID COMPUTING.

Single Processor	Grid Computing (10 processors)
110 mins (Including 13 mins of image tree creation, 95 mins of ray tree creation and 2 mins of overhead).	27mins (including 13 mins of image tree creation and a maximum of 10 mins of ray tree creation for each sub processing task and 4 mins of overhead)

IV. ROUTE COMPARISION OF WITH MEASUREMENTS

The ray tracing model is now used to perform path loss prediction comparisons with outdoor SIMO measurements in the city of Bristol. The transmitter is mounted on a building top (approx. 30m from the ground) and 400 receiver points are placed along the three routes shown in Figure 7 (approx. 1.7m from the ground). An eight element ULA is used at the transmitter as (shown in Figure 8) and an omni antenna is used at the receiver. Reflection is performed up to the fourth order, diffraction up to the second order, and terrain scattering is also considered. The bandwidth is 20MHz at a centre frequency of 1.92GHz, with a frequency resolution of 156.25 kHz. The database used in the ray tracing model consists of a 1km by 1km area with 995 buildings, 174 foliage objects, 12,495 building polygons, 12,495 building roof top edges, 7,921 building vertical edges and 14,046 terrain pixels (sampled every 10m).

Figure 7. Trial map of 1km by 1km in the city of Bristol

Figure 8. Eight element ULA antenna.

The results are shown in Figure 9 and summarized in Table 3. The prediction results agree well with the measurement data, with a mean error of 3-4 dB. The correlation of each

transmission coefficient plays an important role in the capacity calculation of MIMO systems. Although the trial studied here only provides (8x1) SIMO results, a study of the complex correlation coefficient between each transmission sub channel provides a basis for the validation and implementation of MIMO systems. There are 8 sub element channels giving an 8x8 symmetrical correlation coefficient matrix C_M for each receiver point. A correlation coefficient error matrix C_{EM} of (8x8) is calculated for each receiver point such that C_{EM} is equal to the absolute difference of the measured C_M and the modelled C_M from the ray tracing output.

TABLE III. STATISTICAL SUMMARY OF PATH LOSS PREDICTION COMPARISON.

Mean Error (dB)	3.57
Std. Deviation (dB)	3.19
Correlation	0.85

TABLE IV. STATISTICAL SUMMARY FOR CORRELATION COEFFICIENT ERROR MATRIX C_{EM}.

Mean Error (dB)	0.08
Std. Deviation (dB)	0.18

Table 4 shows a statistical summary of C_{EM}. It can be seen that the ray tracing model produces a low mean correlation error between transmission sub channels and this promises accurate MIMO channel predictions.

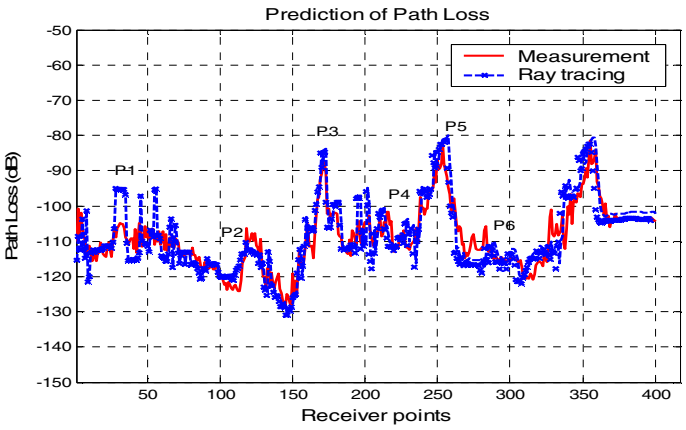

Figure 9. Path loss prediction comparison of the ray tracing output with measurement.

V. MIMO CAPABILITY

The model is able to perform ray tracing using multiple antenna elements at the transmitter and receiver. The program also provides an effective data structure to store the ray tracing results for MIMO analysis. Figure 10 shows an example of the instantaneous normalized MIMO capacity for a 4x4 ULA MIMO system, with a signal-to-noise ratio of 20dB assumed at each receiver location. A 20 MHz bandwidth, a centre frequency of 5.2GHz, and a frequency resolution of 156.25kHz are assumed. A 4-element patch array antenna is used at the transmitter and a 4-element monopole antenna is used at the receiver.

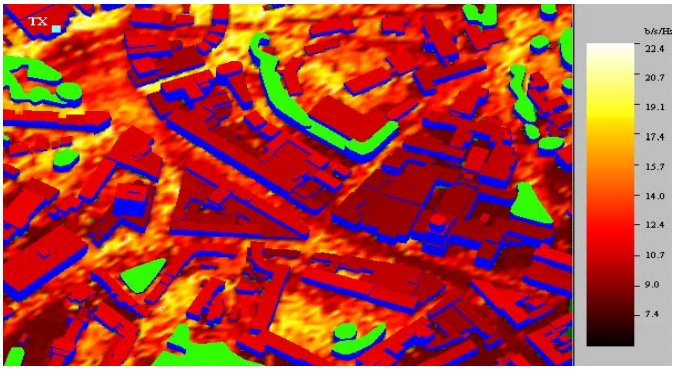

Figure 10. 4x4 ULA instantaneous MIMO capacity prediction.

VI. CONCLUSION

An advanced deterministic ray tracing model has been developed that combines various sophisticated optimization techniques to accelerate the ray path finding process. These features enable the model to perform fast propagation analysis in a complicated outdoor microcellular propagation environment. These features are especially important in the case of MIMO applications where exhaustive ray tracing is performed for all antenna element pairs. Results show a good agreement with measurements for the prediction of path loss and demonstrate the potential for high quality MIMO analysis.

ACKNOWLEDGEMENT

This work was performed under the IST-2001-32549 ROMANTIK project. The authors would like to thank Prof. Mark Beach, Mythri Hunukumbure and Sze Ern Foo for performing the channel measurements used in this study.

REFERENCES

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Techn. J.*, pp. 41–59, Autumn 1996.

[2] G.E. Athanasiadou, A.R.Nix and J.P.McGeehan, "A microcellular ray-tracing propagation model and evaluation of its narrow-band and wide-band predictions," IEEE Journal on Selected Areas in Communications, vol 18., No.3, pp.322- 335, March 2000.

[3] D. A. McNamara, C. W. I. Pistorius, and J. A. G. Malherbe, *Introduction to the Uniform Geometrical Theory of Diffraction. Norwood, MA: Artech House, 1990.*

[4] T.Brook, P.F.Driessen, R.L.Kirlin, "Propagation measurements using synthetic aperture radar techniques", Vehicular Technology Conference 1996, IEEE 46th,vol 3, pp. 1633 – 1637, 28 April-1 May 1996.

[5] R.Hoppe, P.Wertz, G.Wolfle, and F.M. Landstorfer, "Fast and enhanced ray optical propagation modeling for radop network planning in urban and indoor scenarios," *MPRG Wireless Personal Communications Symposium 2000*, Blacksburg (Virginia USA), 9-14 April 2000.

[6] F.Aguado Agelet, F.Perez Fontan, and A.Formella, "Fast Ray Tracing for Microcellular and Indoor Environments," *IEEE Transactions on Magnetics*, 33(2):1484-1487, March 1997.

[7] R.P.Torres, L.Valle, M.Domingo, S.Loredo, "An efficient ray-tracing method for radiopropagation based on the modified BSP algorithm," IEEE Vehicular Tech. Conf., Amsterdam, Sept 1999.

[8] Mark Deloura, *Game Programming Gems*. Massachusetts,USA:Charles River Media, 2000.

[9] Adam Hoult and Gary Simmons. *Engine Programming with BSP Tree.* URL: http://www.gameinstitute.com [06-May-2004].